<u>REMARKS</u>

I.   INTRODUCTION

In response to the Office Action dated March 4, 2004, no claims have been canceled, amended or added. Claims 1-57 remain in the application. Entry of the following remarks, and re-consideration of the application, is requested.

II.   PRIOR ART REJECTIONS

A.   <u>The Office Action Rejections</u>

In paragraphs (2)-(3) of the Office Action, claims 1, 20, and 39 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bayer et al., U.S. Patent No. 5,202,987 (Bayer) in view of Tsuchida et al., U.S. Patent No. 6,026,394 (Tsuchida). In paragraph (5) of the Office Action, claims 1, 20, and 39 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya et al., U.S. Patent No. 5,797,000 (Bhattacharya). In paragraph (7) of the Office Action, claims 2-3, 21-22, and 40-41 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya as applied to claims 1, 20, and 39, in view of Hintz et al., U.S. Patent No. 5,222,235 (Hintz). In paragraph (11) of the Office Action, claims 4-6, 23-25, and 42-44 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya as applied to claims 1, 20, and 39 in view of Bordonaro et al., U.S. Patent No. 5,307,485 (Bordonaro). In paragraph (16) of the Office Action, claims 7-11, 26-30, and 45-49 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya as applied to claims 1, 20, and 39 in view of an "Office Notice" (ON). However, in paragraph (18) of the Office Action, claims 12-19, 31-38, and 50-57 were indicated as being allowable if rewritten in independent form to include the base claim and any intervening claims.

Applicants' attorney acknowledges the indication of allowable claims, but respectfully traverse these rejections.

B.   <u>Applicants' Independent Claims</u>

Applicants' independent claims 1, 16 and 30 are directed to loading data into a data store connected to a computer. Independent claim 1 is representative and comprises the steps of:

identifying memory constraints;

identifying processing capabilities; and

determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities.

-12-

G&C 30571.279-US-01

C.   The Bayer Reference

Bayer describes a high flow-rate synchronizer/scheduler apparatus for a mutiprocessor
system during program run-time, which comprises a connection matrix for monitoring and detecting
computational tasks which are allowed for execution containing a task map and a network of nodes
for distributing to the processors information or computational tasks detected to be enabled by the
connection matrix. The network of nodes possesses the capability of decomposing information on a
pack of allocated computational tasks into messages of finer sub-packs to be sent toward the
processors, as well as the capability of unifying packs of information on termination of
computational tasks into a more comprehensive pack. A method of performing the
synchronization/scheduling in a multiprocessor system of this apparatus is also described.

D.   The Tsuchida Reference

Tsuchida describes a database management system for executing database operations in
parallel by a plurality of nodes and a query processing method. The database management system
contains a decision management node for deciding a distribution node for retrieving information so
as to analyze a query received from an application program, generate a processing procedure for
processing the query, and execute the process, and a join node for sorting, merging, and joining the
information retrieved by the distribution node. When the query process is executed, the distribution
node decided by the decision management node retrieves the information to be processed and the
join node decided by the decision management node also obtains the result for the query from the
retrieved information. The query result is outputted from an output node and transferred to the
application program.

E.   The Bhattacharya Reference

Bhattacharya describes a method of performing a parallel join operation on a pair of
relations R1 and R2 in a system containing P processors organized into Q clusters of P/Q
processors each. The system contains disk storage for each cluster, shared by the processors of that
cluster, together with a shared intermediate memory (SIM) accessible by all processors. The relations
R1 and R2 to be joined are first sorted on the join column. The underlying domain of the join
column is then partitioned into P ranges of equal size. Each range is further divided into M
subranges of progressively decreasing size to create MP tasks $T_{m,p}$, the subranges of a given

-13-

G&C 30571.279-US-01

range being so sized relative to one another that the estimated completion time for task T.sub.m,p is a predetermined fraction that of task T.sub.m-1,p. Tasks T.sub.m,p with larger time estimates are assigned (and the corresponding tuples shipped) to the cluster to which processor p belongs, while tasks with smaller time estimates are assigned to the SIM, which is regarded as a universal cluster (cluster 0). The actual task-to-processor assignments are determined dynamically during the join phase in accordance with the dynamic longest processing time first (DLPT) algorithm. Each processor within a cluster picks its next task at any given decision point to be the one with the largest time estimate which is owned by that cluster or by cluster 0.

F.   The Hintz Reference

Hintz describes a reorganization method of DB2 data files exploring parallel processing, and asynchronous I/O to a great extent. It includes means to estimate an optimum configuration of system resources, such as storage devices (DASD devices), memory, and CPUs, etc, during reorganizations. The method mainly consists of four components, (1) concurrent indexing, (2) concurrent unloading of data file partitions, (3) efficient reloading of DB2 data pages and DB2 space maps, and (4) means to reduce access constraints to the DB2 recovery table.

G.   The Bordonaro Reference

Bordonaro describes a system and method for merging a plurality of sorted lists using multiple processors having access to a common memory in which N sorted lists which may exceed the capacity of the common memory are merged in a parallel environment. Sorted lists from a storage device are loaded into common memory and are divided into a number of tasks equal to the number of available processors. The records assigned to each task are separately sorted, and used to form a single sorted list. A multi-processing environment takes advantage of its organization during the creation of the tasks, as well as during the actual sorting of the tasks.

H.   Applicants' Claimed Invention Is Patentable Over The References

Applicants' attorney respectfully submits that Applicants' claimed invention is patentable over the references. Specifically, Applicants' attorney asserts that the references do not teach or suggest the limitations recited in Applicants' independent claims 1, 20 and 39.

With regard to the rejections based on Bayer and Tsuchida, the Office Action states the following:

-14-

3.　　　Claims 1, 20, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bayer et al. (US Pat 5,202,987, hereinafter Bayer) in view of Tsuchida et al. (US Pat 6,026,394, hereinafter Tsuchida).

4.　　　Regarding claims 1, 20, and 39, Bayer et al. disclose a method of loading data into a data store connected to a computer, the method comprising the steps of:

identifying memory constraints (col. 1, lines 13 - 15, memory and processors are operations bottleneck and col. 5, lines 52 - 56, memory is constrained or limited through factors such as physical shared storage, network access or processor distribution, and common memory space);

identifying processing capabilities (col. 1, lines 17 - 27, synchronization activities are controlled by algorithm, which depends on processing power and col. 5, lines 24 - 31 requires the number of processors and capabilities of each processor, which entail processing capabilities); and

determining a number of load (col. 14, lines 25 - 33, loading capacity being part of task map) to be started in parallel based on the identified memory constraints and processing capabilities (col. 7, lines 9 -1).

Although Bayer disclose the sort process being a mere tasks allocation to the processors (col. 1 , lines 44 - 50), Tsuchida has nevertheless further detailed the sort feature, which includes the step of determining a number of sort processes (col. 8, lines 50 - 51 disclose the fact that the sorting process depends on the number of node for join process. Col. 7, lines 54 - 57 show that the number of join nodes for performing merge process can be determined. Hence, number of sort processes is a known quantity).

It is considered obvious to one of ordinary skill in the art, at the time the invention was made, to combine the sorting feature shown by Tsuchida to the invention of Bayer so that sort processing time, which is a factor in load balancing processes can be determined as part of system characteristics and optimization purposes (Tsuchida: col. 7, lines 58 - col. 8, line 35). Note that the sort steps shown by Tsuchida are also parallel processes as claimed in the application (fig. 3, parallel pipeline operation).

The Office Action also states the following:

Response to Arguments
19.　　　Applicant's arguments filed 12/30/03 have been fully considered but they are not persuasive for the reasons set forth below.

20.　　　In response to applicants' remarks, page 20, lines 1 - 3, and page 22, 4th paragraph, regarding Bayer, Tsuchida and Bhattacharya references do not teach or suggest the limitation "determining a number of load and sort processes to be started parallel based on the identified memory constraints and processing capabilities", this has already been addressed as stated in the rejection above.

Additionally, Tsuchida's fig. 2 teaches a plurality of processors implementing parallel operations in a database management system. Fig 3 teaches management node 12 determines the distribution process with the retrieval data (loading process) by execution on the basis of number of load and sort process (col. 10, line 60 - col.

-15-

G&C 30571.279-US-01

11, line 16). Tsuchida also teaches of dynamic optimization process to produce an optimal result within the system processing capabilities by analyzing the processing procedures that are applicable based on the constraints (fig. 11f, 12b) including calculating the processing time considering each system characteristic (col. 3, lines 6 - 47, col. 12, lines 9 - 35).

Applicants' attorney disagrees. The cited portions of these references do not teach or suggest the limitation "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities."

For example, the cited portions are set forth below:

Bayer: Col. 1, lines 13-27
The coordination of multiple operations in shared memory multiprocessors often constitutes a substantial performance bottleneck. Process synchronization and scheduling are generally performed by software, and managed via shared memory. Execution of parallel programs on a shared-memory, speedup-oriented multiprocessor necessitates a means for synchronizing the activities of the individual processors. This necessity arises due to precedence constraints within algorithms: When one computation is dependent upon the result of other computations, it must not commence before they finish. In the general case, such constraints are projected onto an algorithm's parallel decomposition, and reflected as precedence relations among its execution threads.

Bayer: Col. 5, lines 24-31 (actually, 24-44)
In addition to a task map, the synchronizer/scheduler is supplied with the system configuration data. This includes such details as the number of processors, the capabilities of each processor (if processors are not a-priori identical), etc.
Given a set of enabled tasks, as well as processor availability data, the synchronizer/scheduler then performs scheduling of those tasks. Any non-random scheduling policy must rely upon some heuristics: Even when task execution times are known in advance, finding an optimal schedule for a program represented as a dependency graph is an NP-complete problem. Most scheduling heuristics are bases on the critical path method, and thereby belong to the class of list scheduling policies; i.e., policies that rely on a list of fixed task priorities. List scheduling can be supported by the inventive scheme described herein, by embedding task priorities in the task map load-module submitted to the synchronizer/scheduler. Whenever an allocation takes place, the allocated tasks are those which have highest priorities amongst the current selection of enabled tasks.

Bayer: Col. 14, lines 25-33
Characterizing Parameters
The parameters characterizing a specific synchronizer/scheduler can now be summarized:
Loading Capacity:

-16-

The maximal size of a task map which can be loaded. This parameter is express d in terms of quantity of tasks, and/or in terms of quantity of dependency connections.

Bayer: Col. 7, lines 9-11 (actually, 9-13)

The multiprocessor architecture is illustrated in FIG. 1. As can be seen, the parallel operation coordination subsystem (synchronizer/scheduler 10) forms an appendage to a conventional configuration of a shared-memory 12 and processors 14.

Tsuchida: Col. 8, lines 50-51 (actually, 44-65)

FIG. 8 is a schematic view of the tuning by the slot sort preprocessing. In the same way as in FIG. 7, it is assumed that the data retrieval/distribution process is executed in the nodes #1 to #8 and the processing time in each node is the one shown at each of the numbers 300 to 305. The processing time in each node varies with the number of data in each table. The slot sorting process is set so as to be executed by the nodes for join process. When the processing time varies with each node, the processing procedure for transferring the slot sorting process to the nodes for data retrieval/distribution is considered. For example, in a node where the data retrieval/distribution process is expected to end earlier as slot sort preprocessing, the slot sorting process is executed as shown at 306 to 309. By performing the slot sort preprocessing in this manner, the slot sort processing time by the nodes for join process can be reduced to about the value shown at 312. Using the reduced processing time shown at 311, the N-way merge process is transferred. This is nothing but extension of the run length of the slot sorting process. By doing this, the time 320 required for the N-way merge process can be reduced and as a result, the total response time can be reduced.

Tsuchida: Col. 7, line 54 – col. 8, line 35

Next, the method for deciding the number of join nodes for performing the N-way merge process will be explained with reference to FIG. 7. FIG. 7 is a schematic view for explaining the decision method for the number of join nodes. In FIG. 7, graphs of the phases of parallel join process explained in FIG. 3 and of the processing time of each process are made and laid out according to the parallel pipeline operation explained in FIG. 4. In FIG. 7, it is assumed that the data retrieval/distribution process is executed in the nodes #1 to #8 and the processing time in each node is the one shown at each of the numbers 300 to 305. In this example, the processing time 304 in the node #5 is the maximum processing time. The slot sorting processing time can be driven from the number of nodes for join process N, predetermined system characteristics (CPU performance, disk unit performance, etc.), and database operation method. The performance characteristic (processing time Es) of the slot sorting process can be obtained generally from the following expression.

$$Es=a/N+b*N+c \quad (1)$$

The N-way merge processing time (Em) and join processing time (Ej) also can be obtained from the following expressions.

$$Em=d/N+e*N+f \quad (2)$$
$$Ej=g/N+h*N+i \quad (3)$$

-17-

where, symbols a, d, and g indicate constants which are decided from system characteristics such as the number of rows, the number of pages, each operation unit time, and output time. Symbols b, e, and h are constants which are decided from system characteristics such as the communication time, and c, f, and i are constants which are decided from the other system characteristics

According to this embodiment, to maximize the effect of the pipeline process, the number of nodes for join process is obtained as the number of assigned join nodes 350 so that the performance characteristic Es of the slot sorting process becomes equal to the maximum processing time 304. When the number of assigned join nodes 350 is determined, the N-way merge processing time 320 and join processing time 330 can be estimated from the equations (2) and (3). The total of these processing times is the total processing time for a query. By deciding the number of join nodes in this manner and merging the data distributed in the data retrieval/distribution process successively and processing them simultaneously, the total processing time (response time from querying to output) can be shortened.

Tsuchida: Col. 10, line 60 – col. 11, line 49

FIG. 11(e) shows a detailed flow chart of the process for generation of processing procedure candidates (Step 2213). The process for generation of processing procedure candidates checks whether the table to be accessed for the query is separately stored in a plurality of nodes (Step 22130). When the table is separately stored in a plurality of nodes, the database management system goes to Step 22135. When the table is not separately stored, the process for generation of processing procedure candidates checks whether the sorting process is necessary for executing the query (Step 22131). When the sorting process is necessary for the query process, the database management system goes to Step 22135. When the sorting process is not necessary for the processing procedure candidates, the process for generation of processing procedure candidates checks whether there is only one access path for the table to be accessed for the query (Step 22132). When there is only one access path, the process for generation of processing procedure candidates generates a single processing procedure corresponding to the access path and ends the processing (Step 22133). When there is not only one access path the process for generation of processing procedure candidates generates a plurality of processing procedures corresponding to the access paths and ends the processing (Step 22134). At Step 22135, the process for generation of processing procedure candidates decomposes the query to two-way joins which are joinable. Next, the process for generation of processing procedure candidates generates processing procedure candidates for data read on the basis of the registered access path candidates and processing procedure candidates for data distribution according to the decomposition result at Step 22135 in correspondence with the storing nodes where the table is separately stored. The process for generation of processing procedure candidates also generates processing procedure candidates for slot sorting when the slot sorting process is to be executed in the storing nodes. The process for generation of processing procedure candidates registers the processing procedure consisting of a combination of these processing procedure candidates as a processing procedure candidate in each distribution node (Step 22136). The process for generation of processing procedure candidates registers the processing procedure consisting of a combination of the slot sorting process procedure, N-way merge

-18-

G&C 30571.279-US-01

processing procedure, and join processing procedure as a processing procedure candidate in each join node in correspondence with each join processing node. Then, the process for generation of processing procedure candidates parameterizes the slot sorting run length and the number of merging times (Step 22137). The process for generation of processing procedure candidates registers the requested data output processing procedure to the requested data output node as a processing procedure candidate in the output node (Step 22138). Finally, the process for generation of processing procedure candidates ends the processing when the decomposition results are all evaluated and repeats Step 22135 and the subsequent steps when any decomposition results are not evaluated (Step 22139).

Tsuchida: Col. 3, lines 6-47

Furthermore, the plurality of nodes include at least one decision management node having an analysis means of receiving a query, analyzing the query, and generating the query processing procedure, a decision means of deciding the distribution nodes and join nodes for performing the execution process on the basis of the query analysis result of the above analysis means, and an output means of outputting the result for the query obtained from the join node. The decision means of the decision management node desirably decides the distribution node on the basis of the query analysis result of the analysis means, calculates the expected processing time in the distribution node, and decides the join node on the basis of this processing time.

The decision means distributes retrieval information equally to each join node on the basis of the expected retrieval information amount in the decided distribution node. Each of the distribution nodes decided by the decision means retrieves information from the storage means on the basis of the query analysis result and distributes the information to another node. The join node inputs information distributed from the distribution node one by one and processes each inputted information. The distribution node and join node process information independently. Each of the join nodes sorts information distributed from the distribution node, merges the sorted information when it consists of a plurality of information types, joins a query on the basis of the merged information, and outputs the result for the query obtained from the join node.

To assign retrieval information equally to the join nodes by the decision means in a more desirable form, the decision management node has a storage means of storing column value frequency information relating to the information of the storage means of each node.

According to the query processing method of the present invention, the number of nodes can be decided in correspondence with the database operation which is executed in each node. When there is a scattering in distribution of data, the data is equally distributed to each node, and each database operation to be executed in each node is parameterized, and the expected processing times are equalized. Therefore, the processing time in each node is not biased and the pipeline operation can be performed smoothly.

Tsuchida: Col. 12, lines 9-35 (actually, col. 12, line 9 – col. 13, line 10)

FIG. 12(b) is a flow chart showing the detailed procedure of the process for dynamic optimization (Step 223). The process for dynamic optimization checks

-19-

G&C 30571.279-US-01

whether there is only one processing procedure generated by the process for query.
When there is only one processing procedure, there is no need to execute the
process for dynamic optimization and the database management system goes to the
process for code interpretation execution without doing anything (Step 22300).
When a plurality of processing procedures are generated by the process for query
analysis, the process for dynamic optimization calculates the predicate selectivity
based upon the substituted constant (Step 22301). Then, the process for dynamic
optimization checks whether processing procedure candidates which are executed in
parallel by a plurality of nodes are contained (Step 22302). When no corresponding
processing procedure is contained, the process for dynamic optimization selects the
processing procedure according to the threshold for access path selection and ends
the processing (Step 22313). When a plurality of processing procedures which are
executed in parallel are contained, the process for dynamic optimization inputs the
column value frequency information (the join column value frequency information,
the number of rows and the number of pages in the table which are to be accessed,
etc.) from the dictionary (Step 22303) and calculates the processing time for data
retrieval/distribution as mentioned above by considering each system characteristic
(Step 22304). Then, the process for dynamic optimization decides the number "p" of
nodes to be assigned to the join process from the processing time calculated at Step
22304 and selects the processing procedure "a1" for realizing the process explained
in FIG. 7 from the processing procedure candidates (Step 22305). Next, the process
for dynamic optimization checks whether there is a scattering in the data
retrieval/distribution processing time in the data retrieval/distribution nodes (Step
22306). When there is a scattering in the data retrieval/distribution processing time,
the process for dynamic optimization selects the processing procedure "a2" for
executing the slot sorting process by nodes which can afford to execute the data
retrieval/distribution process among the data retrieval/distribution nodes, that is, for
realizing the process explained in FIG. 8 (Step 22307). The process for dynamic
optimization increases the number "p" of assigned join nodes as much as "alpha"
and selects the processing procedure "a3" for realizing the process explained in FIG.
9 (Step 22308). Furthermore, the process for dynamic optimization compares the
requested data output processing time with the sum of the join processing time and
the last round of N-way merge processing time and when the former is greater than
the latter (Step 22309), selects the processing procedure "a4" for realizing the
process in which the last round of N-way merge process is transferred to the join
process as explained in FIG. 10 (Step 22310). In consideration of the response time,
the load of each node, and the effect on the response performance of other
transactions, the process for dynamic optimization selects the best suited processing
procedure among the processing procedures "a1" to "a4" which are set above (Step
22311). After the processing procedure is selected, the process for dynamic
optimization generates the data distribution information to be used for the data
distribution process on the basis of the column value frequency information (Step
22312). When there is no column value frequency information, the process for
dynamic optimization generates the data distribution information according to the
join column evaluation value of the hash function. Finally, the process for dynamic
optimization decides the processing procedure which is executed finally according to
the threshold for access path selection and the process for dynamic optimization
ends (Step 22313).

The descriptions set forth above do not teach or suggest the limitation "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities."

Instead, Bayer is directed to the scheduling of synchronized tasks, but tasks are assigned to processors based on precedence and availability, wherein one computation is dependent on another, and wherein a processor is allocated a new task immediately after it terminates the previous one. Moreover, the "Loading Capacity" referred to above at col. 14, lines 25-33 of Bayer, as a "Characterizing Parameter," relates to the maximal size of the task map, wherein the task map is a data structure that identifies dependencies between tasks being performed, and is used as indicated to assign tasks to processors based on availability.

Similarly, Tsuchida is directed a parallel processing database management system, wherein the number of nodes for a join process is obtained so that a performance characteristic becomes equal to the maximum processing time, i.e., a decision means determines which (already started) nodes are to be used to perform the query in order to minimize the expected processing time.

As a result, neither of the Bayer or Tsuchida references teach or suggest "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities." Consequently, it cannot be said that the combination of Bayer and Tsuchida teaches or suggests, or renders obvious, the Applicants' independent claims.

With regard to the rejections based on Bhattacharya, the Office Action states the following:

> 5. Claims 1, 20, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bhattacharya et al. (US Pat 5,797,000, hereinafter Bhattacharya).
> 6. As per claims 1, 20, and 39, Bhattachary et al. disclose a method of loading data into a data store connected to a computer, the method comprising the steps of:
> identifying memory constraints (col. 9, lines 6 - 7, memory becomes a constraint as its capacity is a contributing factor and is limited);
> identifying processing capabilities (fig. 1, number of processors p, col. 4, line 41- col. 5, line 8, each processor is assigned with a specific number of tasks, hence indicating each limited capability); and
> determining a number of load (col. 3, lines 1 - 3, join column domain and tuples are the load, which obviously much be known in order for them to be partitioned and transferred among the cluster, col. 3, lines 7 -18), and sort processes (col. 2, line 62 - col 3, line 6 disclose various method of parallel sort process in which merge join is one example. Since the actual tasks assigned to the processors are determined during the join phase, which is part of the sort process as shown above, number of sort processes are hence inherently determined as well) to be

-21-

started in parallel based on the identified memory constraints and processing capabilities (col. 1 , lines 24 - 28, col. 4, line 64 - col. 5, line 8: parallel tasks based on processing capabilities, col. 2, lines 66 - col. 3, line 6: parallel sort processing).

Applicants' attorney disagrees. The cited portions of the reference do not teach or suggest the limitations "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities."

For example, the cited portions are set forth below:

Bhattacharya: Col. 9, lines 6-7 (actually 6-17)

Each processor 104 of the system 100 is allotted an equal portion 1/P of the memory capacity of universal cluster 108. In the initial portion of the transfer phase, for each processor p (104) of the system 100, the tasks T.sub.m,p corresponding to that processor and residing on a particular cluster 102 are transferred from that cluster to the universal cluster 108, beginning with the task T.sub.M,p having the smallest estimated completion time and progressing in order of increasing task size (i.e., decreasing m), until the allotted portion 1/P is filled (step 532). The remaining tasks T.sub.m,p for each processor p (104) are transferred to the cluster 102 owning the processor, unless they are already resident there (step 534).

Bhattacharya: Col. 4, line 41 – col. 5, line 8

Referring to FIG. 1, a multiprocessor system 100 incorporating the present invention includes P processors 104 organized into Q equal-size clusters 102, each cluster containing P/Q processors. Each processor 104 may be either a uniprocessor or a complex of tightly coupled processors (not separately shown) that, for the purposes of task assignment, are regarded as a single processor. Each cluster 102 also includes one or more direct access storage devices (DASD) 106, which are magnetic disk drives in the system 100 shown. Each processor 104 within a cluster 102 can access any storage device 106 in the same cluster, but cannot access any storage device in any other cluster 102. Processors 104 are interconnected to one another as well as to a single intermediate memory (SIM) 108, to which each processor has access. SIM 108 is also referred to herein as the universal cluster, or cluster 0. In addition to the memory 108 and storage devices 106 shown, each processor 104 also has its own main memory (not separately shown). In the case of a processor 104 comprising a tightly coupled processor complex, such main memory would be shared by the processors of the complex. The elements shown in FIG. 1 are conventional in the art, as are the interconnections between these elements.

Processors 104 are used for the concurrent parallel execution of tasks making up database queries, as described below. A query may originate either from one of the processors 104 or from a separate front-end query processor as described in the concurrently filed application of T. Borden et al., Ser. No. 08/148,091, now U.S. Pat. No. 5,495,606. As further described in that application, within each cluster 102 the query splitting and scheduling steps described below may be performed by an additional processor or processors (not shown) similar to processors 104; such additional processors would not be counted among the P/Q processors 104 per complex 102 to which tasks are assigned.

-22-

G&C 30571.279-US-01

Bhattacharya: Col. 3, lines 1-3 (actually col. 2, line 66 – col. 3, line 6)

In a parallel sort merge join, the relations to be joined are first sorted, in parallel, within their clusters 102 (FIG. 1). In a naive parallel sort merge join, the underlying join column domain might be partitioned into P ranges of equal size, and the tuples transferred accordingly among the clusters 102. However, given a nonuniform distribution of tuples across the underlying domain, there is no guarantee that the amount of join phase work will be equal.

Bhattacharya: Col. 3, lines 7-18 (actually lines 7-24)

In accordance with the present invention, each of the P ranges is further divided into a relatively small number M of components, creating MP tasks $T.sub.m,p$ in all. These components intentionally have nonequal task time estimates. For example, a reasonable approach would be to partition the tasks so that the estimated completion time of a task $T.sub.m,p$ is half that of the previous task $T.sub.m-1,p$. Assuming that the quadratic output term dominates the task time estimates, this can be done by partitioning the tasks in such a manner that the extent of the range of a given task $T.sub.m,p$ (to which the number of tuples in the task is roughly proportional) is $1/.sqroot.2$ times the number of tuples in task $T.sub.m-1,p$. FIGS. 10A and 10B show an example of such a partitioning. FIG. 10A shows estimated task times as a function of m and p, and FIG. 10B shows actual task times, also as a function of m and p. The latter may be different from the former, and will not be known until the join phase, when the tasks are actually performed.

Bhattacharya: Col. 1, lines 24-28

This invention relates generally to a method of performing a parallel query in a multiprocessor environment and, more particularly, to a method for performing such a query with load balancing in an environment with shared disk clusters, shared intermediate memory or both.

The descriptions set forth above do not teach or suggest the limitation "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities."

Instead, Bhattacharya is directed to a parallel join operation, wherein tasks are assigned to processors based on the partitioning of the domain of the join column into P ranges based on the number of processors and the partitioning of each range into M subranges based on the estimated completion time for the task.

As a result, Bhattacharya does not teach or suggest the limitations "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities." Consequently, it cannot be said that Bhattacharya renders the Applicants' independent claims obvious.

-23-

Hintz, Bordonaro and ON fail to overcome the deficiencies of Bhattacharya. Recall that Hintz was cited only against dependent claims 2-3, 21-22 and 40-41, Bordonaro was cited only against dependent claims 4-6, 23-25 and 42-44, and ON was cited only against dependent claims 7-11, 26-30 and 45-49. Moreover, Hintz was cited only for determining a number of build processes based on the number of sort processes, and for teaching that the number of sort processes does not exceed a number of indexes to be built, Bordonaro was cited only for teaching that the number of load processes does not exceed a number of partitions to be loaded, and that the load and sort processes directly dependent on memory constraints, and ON was cited only for teaching to efficiently utilize all processing capabilities required for the desired task. None of these teachings are relevant to the limitations of Applicants' independent claims.

Thus, Applicants submit that independent claims 1, 20 and 39 are allowable over the references. Further, dependent claims 2-19, 21-38 and 40-57 are submitted to be allowable over the references in the same manner, because they are dependent on independent claims 1 and 12, respectively, and thus contain all the limitations of independent claims 1 and 12. In addition, dependent claims 4-9, 11-25 and 27-44 recite additional novel elements not shown by the references.
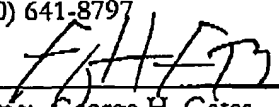
III.   CONCLUSION

In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited. Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicants' undersigned attorney.

Respectfully submitted,

GATES & COOPER LLP
Attorneys for Applicants

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: May 4, 2004

By:_____
Name: George H. Gates
Reg. No.: 33,500

GHG/

-24-

G&C 30571.279-US-01